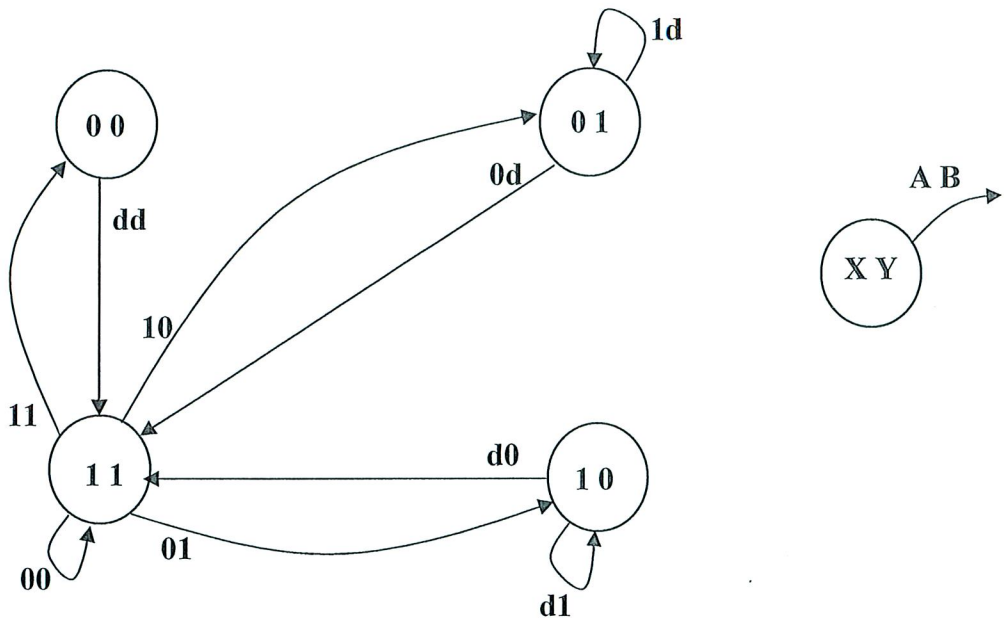
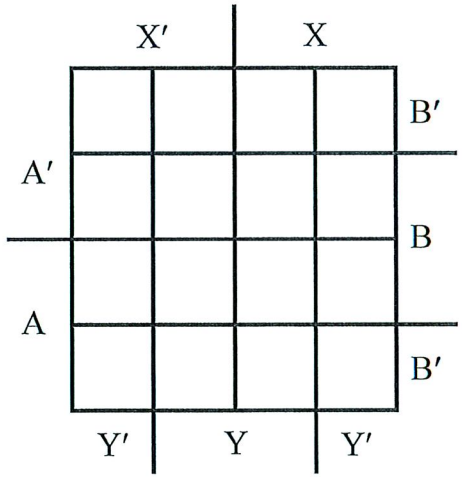
In-Class Homework for Module 3 – No. 1
Monday, March 10, 2014

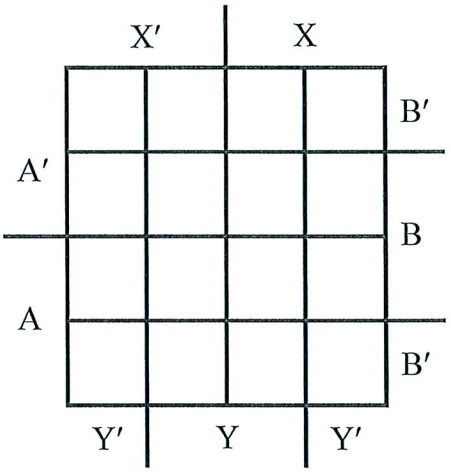
Given the following state transition diagram, determine the ***next state equations*** it represents in ***minimum sum-of-products form.***



X	Y	A	B	X*	Y*
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

X* and Y* are "shorthand" for the next state of X and Y



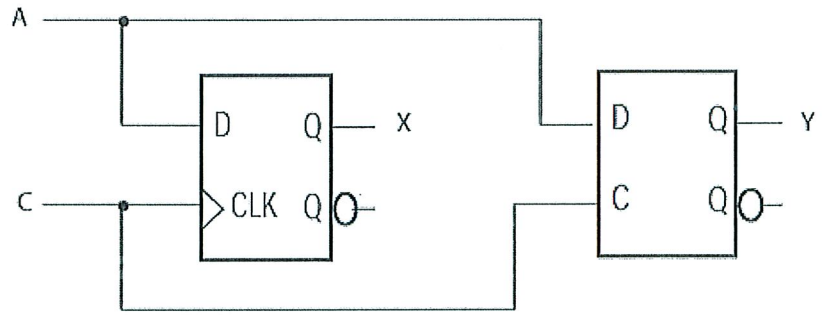


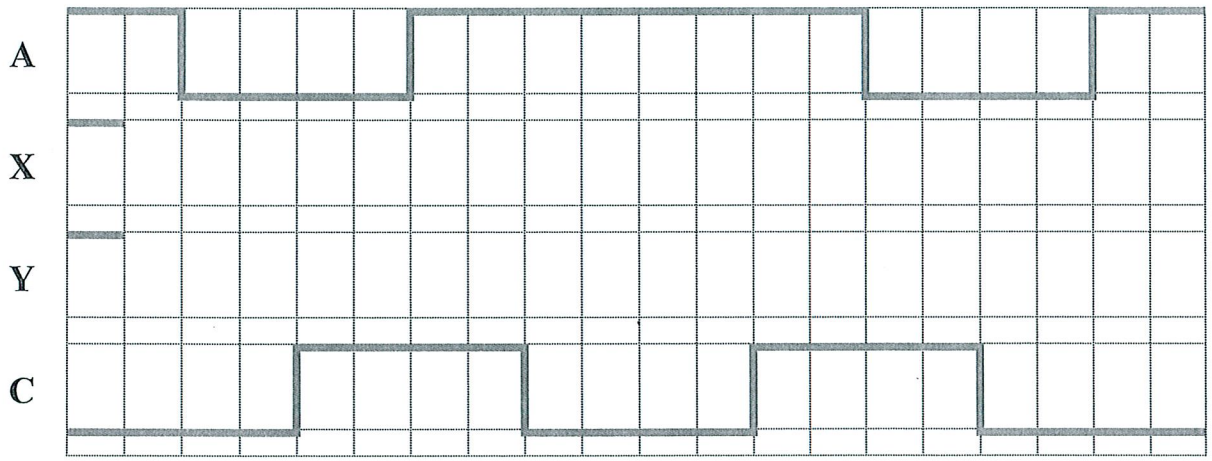
X* = _____

Y* = _____

In-Class Homework for Module 3 – No. 2
Wednesday, March 12, 2014

Assume a positive edge-triggered D flip-flop ("X") and a D latch ("Y") are supplied the signals given on the timing chart, below. Plot the response of each, noting the initial states. Assume the *propagation delays* of the flip-flop and latch are *negligible* relative to the period of "C".

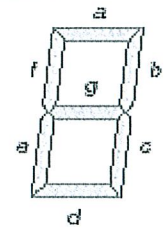




In-Class Homework for Module 3 – No. 3
Monday, March 24, 2014

Design a state machine that outputs the character sequence "AbC" on a 7-segment LED if input control line M=0, and outputs the character sequence "CbS" if M=1.

(a) **Draw a *Moore* model state transition diagram. Note that there are <u>four</u> states, <u>one</u> input (M), and <u>seven</u> active-low outputs (LED segments LA-LG).**



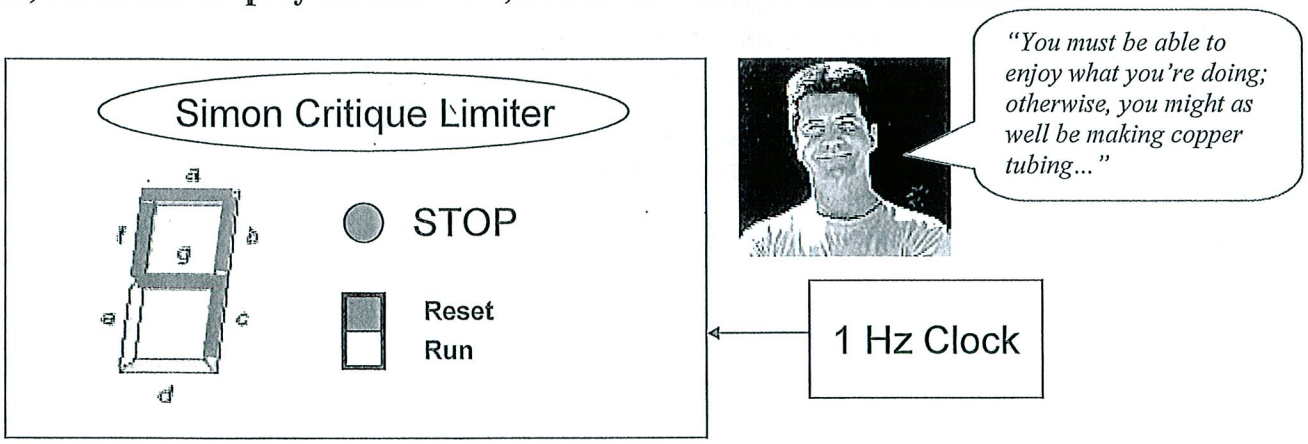
"A" = 1110111
"b" = 0011111
"C" = 1001110
"S" = 1011011

(b) **Complete the ABEL file, below, that implements your design.**

```
MODULE tv_disp
TITLE 'Character Sequence Display for TV'
DECLARATIONS
CLOCK pin;
M pin;     " mode control
Q1..Q0 pin 22..23 istype 'reg';
!LA,!LB,!LC,!LD,!LE,!LF,!LG pin 14..20 istype 'com'; " 7-segment display

TRUTH_TABLE ([Q1, Q0] -> [LA, LB, LC, LD, LE, LF, LG])
             [__, __] -> [ 1,  1,  1,  0,  1,  1,  1];  " A
             [__, __] -> [ 0,  0,  1,  1,  1,  1,  1];  " b
             [__, __] -> [ 1,  0,  0,  1,  1,  1,  0];  " C
             [__, __] -> [ 1,  0,  1,  1,  0,  1,  1];  " S

TRUTH_TABLE ([Q1, Q0,  M] :> [Q1, Q0])
             [ 0,  0,  0] :> [__, __];
             [ 0,  0,  1] :> [__, __];
             [ 0,  1,  0] :> [__, __];
             [ 0,  1,  1] :> [__, __];
             [ 1,  0,  0] :> [__, __];
             [ 1,  0,  1] :> [__, __];
             [ 1,  1,  0] :> [__, __];
             [ 1,  1,  1] :> [__, __];
EQUATIONS

_____________________________________________________________ ;

END
```

In-Class Homework for Module 3 – No. 4
Wednesday, March 26, 2014

America voted, and decided that it would be a good idea if the amount of time allowed for Simon to "critique" each contestant on *American Digital Idol* was limited to nine seconds. This counter should *stop* once the display reaches "0" and, after being *asynchronously reset*, should restart the "down-count" sequence at "9" (note that as long as the *asynchronous reset* is asserted, the display should remain at "9"). Also, when the display reaches "0", a "STOP" LED should be illuminated.



Draw a *Moore* model state transition diagram for the Simon Critique Limiter (SCL). Note that there are <u>four</u> state variables (Q3...Q0) and <u>eight</u> outputs: LED segments "A" through "G" and the "STOP" indicator. Note also that the STOP signal, when negated, *enables* the state machine; when it is asserted, the STOP signal *disables* the state machine. Recall that the *asynchronous reset* will clear all flip-flops to "zero", and it is in this state ("0000") that your 7-segment display should be "9".

Complete the ABEL file, below, that implements your SCL.

```
MODULE simon_sez
TITLE 'Simon Critique Limiter'

DECLARATIONS
CLOCK pin; " connected to a 1 Hz clocking source
ARS   pin; " asynchronous reset

!A    pin istype 'com'; " display segment A
!B    pin istype 'com'; " display segment B
!C    pin istype 'com'; " display segment C
!D    pin istype 'com'; " display segment D
!E    pin istype 'com'; " display segment E
!F    pin istype 'com'; " display segment F
!G    pin istype 'com'; " display segment G
!STOP pin istype 'com'; " STOP indicator

Q3..Q0 pin istype 'reg'; " state machine flip-flops

EQUATIONS

Q0 := ________________________________________________________________;

Q1 := ________________________________________________________________;

Q2 := ________________________________________________________________;

Q3 := ________________________________________________________________;

STOP = _______________________________________________________________;


" asynchronous reset and clocking equations

________________________________________________________________________;

________________________________________________________________________;


TRUTH_TABLE ([Q3,Q2,Q1,Q0]->[ A, B, C, D, E, F, G])
             [ 1, 0, 0, 1]->[ 1, 1, 1, 1, 1, 1, 0]; " 0
             [ 1, 0, 0, 0]->[ 0, 1, 1, 0, 0, 0, 0]; " 1
             [ 0, 1, 1, 1]->[ 1, 1, 0, 1, 1, 0, 1]; " 2
             [ 0, 1, 1, 0]->[ 1, 1, 1, 1, 0, 0, 1]; " 3
             [ 0, 1, 0, 1]->[ 0, 1, 1, 0, 0, 1, 1]; " 4
             [ 0, 1, 0, 0]->[ 1, 0, 1, 1, 0, 1, 1]; " 5
             [ 0, 0, 1, 1]->[ 1, 0, 1, 1, 1, 1, 1]; " 6
             [ 0, 0, 1, 0]->[ 1, 1, 1, 0, 0, 0, 0]; " 7
             [ 0, 0, 0, 1]->[ 1, 1, 1, 1, 1, 1, 1]; " 8
             [ 0, 0, 0, 0]->[ 1, 1, 1, 1, 0, 1, 1]; " 9  reset state

END
```

In-Class Homework for Module 3 – No. 5
Monday, March 31, 2014

Complete the ABEL file that implements a 3-bit programmable binary UP counter, i.e., a counter that counts up to the value D2D1D0 (entered on DIP switches) and resets to zero. The current state of the counter is displayed as a BCD digit on a 7-segment common anode display.

```
MODULE prog_cnt

TITLE 'Programmable 3-bit Binary UP Counter with 7-segment Display'

DECLARATIONS

CLOCK pin;  " CLOCK input
D0,D1,D2 pin;  " data inputs
ARESET pin;  " asynchronous reset
!LA,!LB,!LC,!LD,!LE,!LF,!LG pin istype 'com';  " LED segments
Q0..Q2 pin istype 'reg';
SRESET pin istype 'com';

EQUATIONS

SRESET = ______________________________________________________;

Q0 := ________________________________________________________;

Q1 := ________________________________________________________;

Q2 := ________________________________________________________;

[Q2,Q1,Q0].CLK = _____________________________________________;

[Q2,Q1,Q0].AR = ______________________________________________;


TRUTH_TABLE([Q2,Q1,Q0] -> [LA,LB,LC,LD,LE,LF,LG])
            [ 0, 0, 0] -> [ 1, 1, 1, 1, 1, 1, 0]; "0
            [ 0, 0, 1] -> [ 0, 1, 1, 0, 0, 0, 0]; "1
            [ 0, 1, 0] -> [ 1, 1, 0, 1, 1, 0, 1]; "2
            [ 0, 1, 1] -> [ 1, 1, 1, 1, 0, 0, 1]; "3
            [ 1, 0, 0] -> [ 0, 1, 1, 0, 0, 1, 1]; "4
            [ 1, 0, 1] -> [ 1, 0, 1, 1, 0, 1, 0]; "5
            [ 1, 1, 0] -> [ 1, 0, 1, 1, 1, 1, 0]; "6
            [ 1, 1, 1] -> [ 1, 1, 1, 0, 0, 0, 0]; "7

END
```